Software Architectures

Performance Fundamentals

Duc-Man Nguyen, Ph.D
mannd@duytan.edu.vn
0904 235 945

1

Today

- Definition of Performance
- Performance Quality Attribute Scenarios
- Analyzing Performance
- Performance Tactics

2

Review

1. Which is the first step in the software development life cycle ?

a) Analysis

b) Design

c) Problem/Opportunity Identification

d) Development and Documentation

3. In Design phase, which is the primary area of concern ?

a) Architecture

b) Data

c) Interface

d) All of the mentioned

Review

3. The importance of software design can be summarized in a single word which is:

a) Efficiency

b) Accuracy

c) Quality

d) Complexity

4. Which is not a step of Requirement Engineering?

a) Requirements elicitation

b) Requirements analysis

c) Requirements design

d) Requirements documentation

It's Got To Be "Fast" … Right?

- What do we mean by "performance"?

Performance Defined

- Performance can mean different things depending on the context
- Typically people mean one or more of the following:
  - Latency
  - Throughput
  - Predictability
  - Utilization

Performance Defined

- Performance can mean different things depending on the context
- Typically people mean one or more of the following:
  - Latency
  - Throughput
  - Predictability
  - Utilization

Latency

- Latency is the period of delay between some input or action and some desired result
- This is a concern for many systems
  - The nature of the concern is different from one system to another, however
- One factor to consider is what happens if the desired latency is not reached
  - Is there still value in completing the task after the "deadline" is passed?
- Another word for value is "utility"

When Is This A Concern?

- Latency is a concern for many systems
  - Can you give some examples?

Train Control System

- If you have a train control system what is the “utility” if the track switches after the deadline?

Online Retailer

- How does the utility degrade if you're buying a book?

Worst Case vs Average Case

- For "hard real-time systems" we are worried about worst-case latency
  - These are systems where the utility drops off very quickly after the deadline
- In other systems we are worried about average case response times
  - Many user facing systems (e.g. Google) are concerned with average case response times
- Systems that can be characterized as "soft real-time" are in the middle
  - These systems need to complete some percentage of the results within a specified time limit (often penalties associated with terms)

Performance Defined

- Performance can mean different things depending on the context
- Typically people mean one or more of the following:
  - Latency
  - Throughput
  - Predictability
  - Utilization

Throughput

- Throughput is concerned with the number of tasks executed per unit time
- Possible measures for throughput include:
  - Transactions per second
  - Page views per second
  - …

Throughput Examples

- Many systems today are concerned with throughput
  - Think of the systems that you use (e.g. Facebook, Twitter, Snap Chat, …)
  - Also storage solutions, fraud detection systems, ad exchanges, …
- We are in the age of “Internet Scale” systems where the volume of data and number of users is ever increasing

Performance Defined

- Performance can mean different things depending on the context
- Typically people mean one or more of the following:
  - Latency
  - Throughput
  - Predictability
  - Utilization

Predictability

- Predictability talks about the variation between executions
  - This is sometimes called “jitter”
- This refers to how deterministic the execution time is
- This is important when trying to schedule tasks
  - We will talk about scheduling in a bit

Performance Defined

- Performance can mean different things depending on the context
- Typically people mean one or more of the following:
  - Latency
  - Throughput
  - Predictability
  - Utilization

Resource Utilization

- Another important consideration that sometimes comes under the performance heading is “resource utilization”
  - This could have to do with the network, CPU, power consumption, memory, or other resource

Today

- Definition of Performance
- Performance Quality Attribute Scenarios
- Analyzing Performance
- Performance Tactics

General Quality Attribute Table

Portion of the Scenario	Possible Values
Source	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event
Artifact	System or one or more (runtime) components of the system
Environment	Operational mode: normal, degraded, peak load, overload
Response	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate

Stimulus: Arrival Patterns

- ***Periodic***: arrives at a regular interval (period)
- ***Stochastic***: arrives in a probabilistic distribution
  - Arrival patterns can be analyzed statistically but not precisely predicted
- ***Sporadic:*** occurs infrequently and without a regular interval

Artifact: System Components

- While it's true that detailed design decisions can impact performance
  - Algorithm design for example
- These are typically downstream decisions
  - They are not architectural concerns

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Artifact: System Components

- Analyzing the performance characteristics of a design relies on knowledge of the structures present at runtime
  - Thus the "artifact stimulated" in a performance scenario should be a runtime component
- Examples include:
  - Database, thread, process, server, …

Environment: Operational Conditions

- As we'll discuss shortly performance analysis relies on knowledge of all system tasks
  - Imagine the information you'd need to answer "how long will it take to check out of the grocery store?"
- The environment describes what else is going on in the system when the stimulus occurs
  - This could be from the demand side (other tasks present)
  - Or from the supply side (state of the units of execution)

Response

- The response describes the system's response to the stimulus
- This does not contain any quantification
  - As a result, if you look at just the stimulus and response, this is just a functional requirement

Response

- The response could reflect the “happy path”
- In some cases it might also indicate a degraded mode of operation
  - This implies that something is not right in the system
  - The condition that results in a degraded mode needs to be reflected in either the stimulus or environment

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Response Measure: Quantification

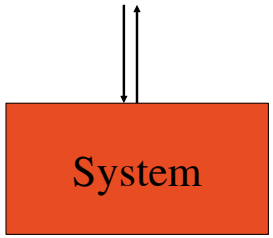
- This is the performance measure that establishes the desired behavior
- The measure should align with the intent
- Think about how the measure would differ between:
  - Worst case response times
  - Average case response times
  - % of missed deadlines

General vs Concrete Scenarios

- The provided table is useful for generating specific (or "concrete") scenarios
- If you know you have a performance concern, the general quality attribute table can help guide the elicitation
- You need to make the scenario specific before it's actionable, however
- You don't need to have a table or explicit identification of the components of the scenario
  - You just need to make sure all the requisite info is present

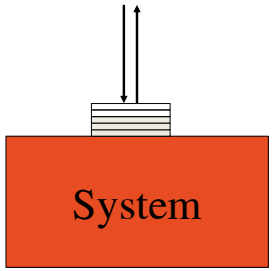
Simple Case

- What contributes to the latency if you have one task in the system and one unit of execution?



Multiple Tasks

- What contributes to latency if you have multiple tasks?



ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Calculating Latency

- In general (we'll ignore communication time for now):

  Latency = Execution time + Wait time

- If you know sufficient information you can build a model to predict latency
  - We won't go into the details of building the model here
  - We will look at the inputs for the model, however

Recall …

- These are the kinds of response measures that could apply to your system
  - Average case response time
  - Worst case response time
  - % of missed deadlines
- The model you'd build depends on what outcome you're interested in
- Average case & % of missed deadlines → queueing model
- Worst case → schedulability model

Queueing Model

- Queueing theory is the mathematical study of queues
  - A queueing model will predict the queue length & wait time
- To build the model, you will need to know:
  - Distribution of arrival time
  - Execution time for each task type
  - Scheduling strategy
  - Number of parallel nodes
- With this information queuing theory will give you a distribution of the expected wait time and queue length

Determining Execution Time

- In order to determine execution time for a task we need to:
  - Understand what the responsibilities are
  - Calculate/estimate the execution time for those responsibilities on a given platform
- This calculation can be based on:
  - Historic data
  - Benchmark data
  - Experiment

Determining Load

- Determining likely load is difficult
  - But important (and often overlooked)
- In order to understand the load you need to know something about the nature of the tasks
  - We can start to think about inbound tasks by looking at the context diagram
  - We can't ignore internally generated tasks, however
- Look at the nature of tasks:
  - Periodic
  - Stochastic
  - Sporadic

Predicting Arrival Distribution

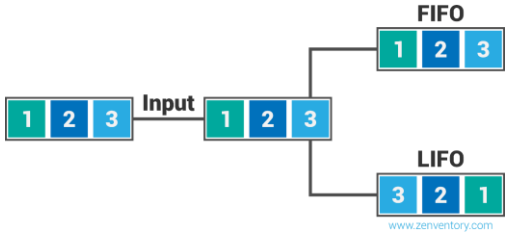
- For stochastic tasks we need to understand the arrival distribution
  - If we are dealing with an existing system, this can come from historic data
- If we don't have historic data we need to make an educated guess
  - Based on business forecasts
  - Maybe benchmark data is available
- The benefit of such models is that's is easy to vary assumptions and see what impact that would have

Scheduling Periodic Tasks

- With periodic tasks you can determine the *schedulability* of your tasks
- A number of scheduling strategies exist
  - Least laxity
  - Rate Monotonic scheduling
  - Shortest deadline first
- Rate Monotonic Scheduling is what's called an "optimal" scheduling strategy
  - If your tasks can be scheduled, they can be scheduled with Rate Monotonic Scheduling

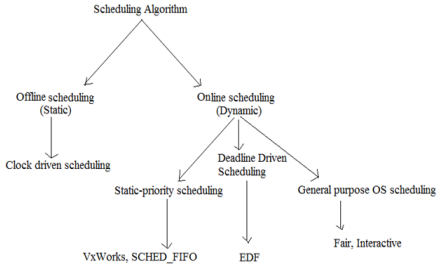
Scheduling Strategy

- This is often realized in the queues
- By default these are often FIFO
- Care needs to be taken when configuring infrastructure
  - You can undermine intended scheduling strategy



Scheduling Strategies

- There are a couple of general categories of scheduling strategies
  - Static
  - Dynamic



Static Scheduling

- With static scheduling priorities are established offline prior to execution
- Can be preemptive or not
- Arrival rates are (typically) known upfront
- (typically) assumes computation time is known and constant

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Dynamic Scheduling

- Priorities are determined at runtime
- Can be preemptive or not
- Priorities change based on changing arrival rates

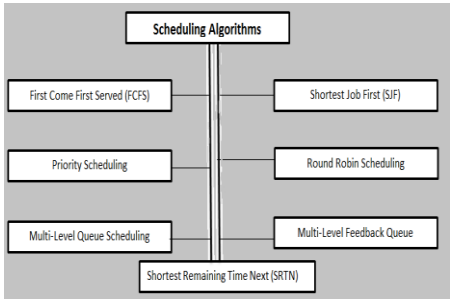
Preemptive vs. Non-preemptive

- Scheduling policies can allow tasks to run to completion before switching
- Or they can “preempt” each other and switch contexts
- This requires a system that supports interrupts

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

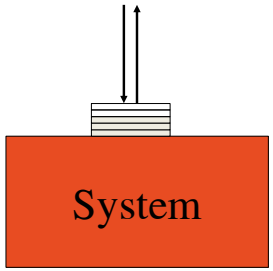
Scheduling Examples

- FIFO
- Round robin
- Shortest process next
- Least laxity
- Earliest deadline first
- Rate monotonic



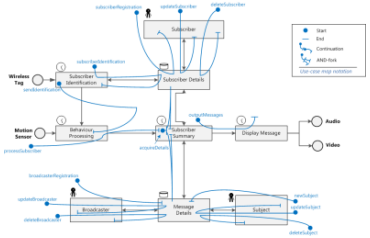
Single Unit of Execution

- So far we've talked about a single unit of execution
  - What happens if the system has multiple units of execution?



Use Case Map

- You need to understand the system elements involved with task execution
  - In order to do this you have to map tasks to units of execution



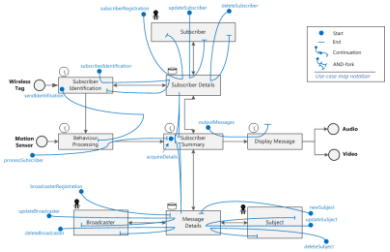
Determining Element Load

- You can determine the wait time for each element by repeating the steps we've talked about
  - Identify expected load for each element
  - Determine execution time for each task type
  - Identify scheduling strategy
  - Use analysis model (e.g. queueing theory) to analyze performance

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Performance Across the System

- We can then look at the performance across our system



Identifying Bottlenecks

- This will allow us to identify bottlenecks in our system
- Bottlenecks are congested areas in the system that contribute significantly to the overall latency
  - They are the "weakest link in the chain"
- These are portions of the system that dominate due to either wait time or execution time
  - These are likely candidates for improvement

Let's Re-iterate

1. We first look at what the load is expected to be
   - What are the tasks that are going to be performed
   - How many users are expected
   - What's the usage patterns
2. We map each of these tasks to a candidate design
   - We need to determine what the units of execution are for each task
   - We estimate execution time for the sub tasks associated with each unit

Let's Re-iterate

1. We calculate wait time at each unit of execution
   - This is based on the arrival distribution of each task, the execution time of each task and the scheduling strategy
2. We identify bottlenecks
   - If we are not going to achieve our deadlines what are the areas to improve
3. We modify our design and repeat

Improvement Options

- If we want to reduce latency our options are:
  - Reducing wait time
  - Reducing execution time

Reducing Wait Time

- Scheduling strategy
- Manage event rate
- Bound execution times
- Bound queue sizes
- Introduce concurrency

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Reducing Wait Time

- Scheduling strategy
- Manage event rate
- Bound execution times
- Bound queue sizes
- Introduce concurrency

Scheduling Strategy

- Again the naïve approach is often FIFO
- We could vary the scheduling strategy to change the wait time for critical tasks
- The different scheduling strategies react differently to sets of tasks
  - They degrade differently
  - They support differing levels of analysis
- Simulation may be a good idea
- Care is required to ensure that the strategy is consistently implemented

Reducing Wait Time

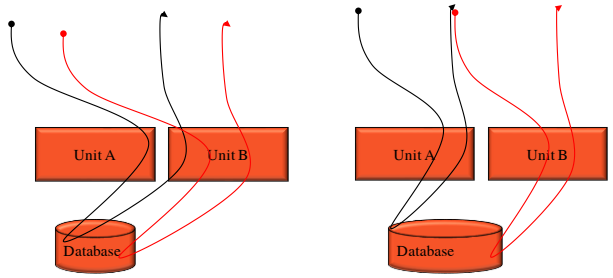
- Scheduling strategy
- Manage event rate
- Bound execution times
- Bound queue sizes
- Introduce concurrency

Manage Event Rate

- In some cases it's possible to manage periodic tasks
  - Perhaps ping/echo, heartbeat, or other periodic task rates can be adjusted
- You can also manage stochastic events in some cases e.g.:
  - By bundling messages
  - Or reassigning responsibility
- As systems reach overload you could selectively shed load as well

Reassigning Responsibility

- If B is overloaded you can reduce responsibilities



Managing Event Rate II

- Synchronization across replicas can be a big source of load
- To minimize this you can
  - Configure “groups” within the cluster to share session state
  - Offload “intra cluster” traffic
  - Bundle messages
  - Reduce the amount of state managed

Reducing Wait Time

- Scheduling strategy
- Manage event rate
- Bound execution times
- Bound queue sizes
- Introduce concurrency

Bound Execution Times

- Another strategy is to bound execution times
- Tasks may “run over” expected execution times
  - This could be due to network delays, database congestion, or other reason
- In order to minimize the impact on other tasks timeouts can be imposed

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Reducing Wait Time

- Scheduling strategy
- Manage event rate
- Bound execution times
- Bound queue sizes
- Introduce concurrency

Bound Queue Sizes

- If queues are infinitely large it's clear not all tasks will be processed in a timely manner
- Limiting queue sizes to those that can be managed will help "throttle" system load

Reducing Wait Time

- Scheduling strategy
- Manage event rate
- Bound execution times
- Bound queue sizes
- Introduce concurrency

Introduce Concurrency

- In areas where bottlenecks may occur concurrency can be introduced to limit wait time
- Typically bottlenecks include:
  - Database connections
  - Waiting for an available thread
  - Web servers
  - Application servers

Concurrent Units of Execution

- Additional bank tellers will reduce your wait time in line at a bank
- Likewise additional units of execution can reduce wait time in your system
- There are several ways this can take place
  - The decision is how you are going to allocate tasks to the units of execution

Reduce Processing Time

- Increase computational efficiency
- Reduce computational overhead
- Increase resources

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Increase Computational Efficiency

- In addition to standard programming practices other options include:
  - Cache data
  - Reduce levels of abstraction
- Several kinds of data caching can be implemented e.g.
  - Session caching
  - Database caching

Reduce Computational Overhead I

- There are many mechanisms that add overhead
- Some of the areas include:
  - Use more efficient means of communication
  - Bundle messages
  - Reduce replication
  - Reduce data caching

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Reduce Communication Overhead

- Communication
  - Many of the communication mechanisms introduce significant overhead
  - In some cases it may be worthwhile to bypass these mechanisms
  - Transactions, groups, and other supports also add overhead and should be used sparingly

Data Caching

- While in some cases caching data can improve performance
- In other cases it can negatively impact performance
  - When replicas are involved
- Reducing the state that is managed by these replicas can improve performance
  - Session and application state replication are configurable items
- Caching can be a real tradeoff – what is the tradeoff? When?

Increase Resources

- More powerful hardware
- Additional NIC cards
  - Separate cluster traffic from external traffic

Summary

- Calculating load (system wide and for each processing unit) is an important first step for determining performance
  - This is often not done
- There are many factors that influence the performance
  - Overhead and wait time are the biggest contributors
- Careful consideration of the mechanisms is important to balance performance concerns with other systemic properties

References

- https://www.youtube.com/watch?v=MSxdNG2d_L4 (Performance Tactic)

First In First Out (FIFO)

- Tasks are processed in the order they arrive
- Non preemptive, a process isn't blocked until it voluntarily gives up the CPU
- Easy to implement
- Short tasks are penalized more heavily
- Favors computationally intensive tasks over interactive tasks
- No way to break out of an infinite loop

Round Robin

- Each task is processed for a period of time before moving to the next task
- Is a preemptive policy
- No penalty ratio between long and short tasks
- Preemption is expensive, however

Shortest Process Next

- The task with the shortest execution time is given the highest priority
- Non preemptive
- Execution time is estimated
- Can lead to starvation
  - Longer running tasks may never run

ĐẠI HỌC DUY TÂN
International School
Khoa Đào tạo Quốc tế
http://kdtqt.duytan.edu.vn

Earliest Deadline First

- Dynamic and preemptive
- Task with the earliest deadline is processed
- Does not degrade well
  - If you're overloaded allot of tasks won't complete

Least Laxity

- Priority is assigned based on the “slack time”
  - This is the amount of time between completion of the job and the deadline
- Dynamic scheduling policy
  - Typically used with aperiodic tasks
- Can be sub optimal
- Degrades more gracefully
  - Can shed tasks with no hope of completing

Rate Monotonic Scheduling

- Static scheduling policy
- The shorter the *period* the higher the priority
  - The period is the arrival rate
- Policy is “optimal”
  - If a set of tasks cannot be scheduled using RMS then it can’t be scheduled with any scheduling strategy
- The utilization is not always optimal, however
  - This is true for fixed-priority scheduling in general
- Harmonic periods will allow you to schedule a set of tasks up to 100% utilization

Configure Session Groups

- In a cluster you can define “Buddy Replication” groups
- These are smaller groups for replicating session state
- Thus session replication won’t occur across all of the nodes
- Reduces the overhead associated with synchronizing session
  - Limits availability for a given session

Offload “Intra-Cluster” Traffic

- You can separate external messages from intra-cluster messages
- This requires duplicate NIC cards
- Intra-cluster traffic is routed to a separate IP address
- This is a means of introducing concurrency (and thus reducing load)

Bundle Messages

- At design time you can determine the granularity of messages
- This will limit the overhead associated with packaging of messages
- There is a tradeoff, however, as you will then queue messages until all are ready
- This is often a configurable item